On the power of alternations in space-bounded computations

AUTHOR(S):

Toda, Seinosuke

CITATION:

Toda, Seinosuke. On the power of alternations in space-bounded computations. 数理解析研究所講究録 1987, 625: 118-131

ISSUE DATE:

1987-05

URL:

http://hdl.handle.net/2433/99959

数理解析研究所講究録
625巻 1987年 118-131

線型領域におけるオルタネーションの能力について

On the power of alternations in space-bounded computations

国文学研究資料館　戸田誠之助　(Seinosuke Toda)

Abstract

One of the most important questions in the theory of computational complexity is whether nondeterministic space-bounded complexity classes are closed under complement. In the case of linear space bound, this can be viewed as a second version of a famous question known as the LBA problem in Kuroda[3], and is a longstanding open question. Futhermore, scince the concept called "alternation" was introduced in Chandra, Kozen and Stockmeyer[1] as a generalization of nondeterminism the same many questions as the above one have arised. A typical one in those questions is whether the linear space-bounded complexity class defined by alternating Turing machines allowing with one alternation is closed under complement. We can view this as a third version of the LBA problem. In this paper, we answer this question, that is, we show that Σ_2SPACE(n) is closed under complement, where Σ_kSPACE(n) denotes a class of languages accepted by linear space-bounded alternating Turing machines whose initial state is existential and which make at most k-1 alternations, for each $k>0$. As an immediate consequence, the alternation linear-space hierarchy collapses to the second level, i.e., Σ_2SPACE(n) = Σ_kSPACE(n) for any $k\geq 2$. By using a usual translational method, these results can be extended to any space bound greater that linear.

1. Preliminaries

We assume that the readers are familiar with the basic concepts from the theories of automata, computability and formal languages. We outline the necessary concepts in this paper.

Let T denote an appropriate finite alphabet. For a word w in T^*, $|w|$ denotes the length of w. The empty word is denoted by λ. For a language $A \subseteq T^*$, A^c denotes its complement(i.e., $A^c = T^* - A$). For a class C of languages, coC denote the class of languages whose complement is in C.

Our models of computation are variations of one-tape Turing machines (see [2]). A <u>one-tape Turing machine</u> (TM for short) has a two-way read-write tape used as both input tape and work tape, and the distinguished three states called the <u>initial</u>, the <u>accepting</u> and the <u>rejecting</u>. An <u>instantaneous description</u> (ID for short) of a TM M on input x is a pair $(q, u\uparrow v)$, where q indicates a state of M, its tape contains a word uv, and the tape head currently reads the leftmost symbol of v. An ID is <u>initial</u> (<u>accepting</u> and <u>rejecting</u>) iff the state of the ID is initial (accepting and rejecting, respectively). For any ID I and J, if M moves from I to J in one step, then this movement is abbriviated by $I \vdash J$. A computation path of M on x is a sequence of IDs I_1, I_2, ..., I_m such that $I_i \vdash I_{i+1}$ for each $1 \leq i \leq m$.

An <u>Alternating Turing machine</u> (ATM for short) is a

generalization of ordinary TMs, described informally as follows. The states of an ATM M are classified into two distinct states called universal states and existential states, respectively. We can view a computation of M on input x as a tree whose nodes are labeled with IDs of M on x. A computation tree of M on x is a tree such that any internal node labeled with a universal (existential) ID is followed by all (resp., one) of successors of that ID, where an ID is universal (existential) if the state of the ID is universal (resp., existential). An accepting computation tree of M on x is a computation tree such that its root node is labeled with the initial ID and each of its leaves is labeled with an accepting ID. M accepts x iff there is an accepting computation tree of M on x.

An ATM is said to be deterministic (abbriviated by DTM) iff its transition function is one-valued. An ATM is said to be nondeterministic (abbriviated by NTM) iff its states are all existential.

Let I and J be any IDs of an ATM M on input x. A move $I \vdash J$ of M on x is called an alternation move iff either I is universal and J is existential, or I is existential and J is universal. Let k be any positive integer. An ATM M is said to be k alternation-bounded iff for each input x, M makes at most k-1 alternation moves in each computation path. A σ_kTM is a k alternation-bounded ATM whose initial state is existential. A

π_kTM is a k alternation-bounded ATM whose initial state is universal. We note that a σ_1TM is an NTM.

Let f be a function on positive integers. An ATM is f(n) space-bounded iff for each input x, M uses at most f(|x|) tape cells. Let S be any space bounds greater than or equal to linear. DSPACE(S(n)) denotes a class of languages accepted by O(S(n)) space-bounded DTMs. NSPACE(S(n)) denotes a class of languages accepted by O(S(n)) space-bounded NTMs. For any positive integer k, Σ_kSPACE(S(n)) denotes a class of languages accepted by O(S(n)) space-bounded σ_kTMs, and Π_kSPACE(S(n)) denotes a class of languages accepted by O(S(n)) space-bounded π_kTMs. We note that NSPACE(S(n))= Σ_1SPACE(S(n)) and coNSPACE(S(n))= Π_1SPACE(S(n)) by definitions.

A function S on natural numbers is called space-constructible if for input 1^n, $1^{S(n)}$ can be constructed by an S(n) space-bounded DTM.

Through the whole of this paper, we assume the following constraints on an ATM M. Without loss of generality, we can assume so.

(1) The input alphabet of M is {0,1}. Hence we consider only languages on $\{0,1\}^*$. The tape alphabet of M may contains any finite number of symbols and includes {0,1,B}, where B denotes the blank symbol contained initially in each tape cell except

input.

(2) Let $S(n) \geq n$ be any space-constructible function and let M be S(n) space-bounded. Then, for each input x, M first constructs $x\#^{S(|x|)-|x|}$ on its tape and M begins its substential computation after this construction. Futhermore, at any time of the substential computation, the length of contents of its tape is always $S(|x|)$ (i.e., M never uses another tape cell except the cells $x\#^{S(|x|)-|x|}$ is written initially and M never writes the blank symbol).

Although the assumption (2) is technical one, it is essential in the proof of main theorem. The assumption (1) is a conventional one.

At the end of this section, we state some basic properties about the above classes without the proof.

Proposition 1.

(1) For any $k>0$, if $\Pi_k SPACE(n) \subseteq \Sigma_k SPACE(n)$, then the equality holds.

(2) If $\Sigma_k SPACE(n) = \Pi_k SPACE(n)$ for some $k>0$, then $\Sigma_k SPACE(n) = \Sigma_i SPACE(n)$ for any $i \geq k$.

(3) For any $k,i>0$, if $\Sigma_k SPACE(n) = \Sigma_i SPACE(n)$, then $\Sigma_k SPACE(S(n)) = \Sigma_i SPACE(S(n))$ for each space-constructible $S(n) \geq n$.

2. Main results

Our main theorem in this paper is as follows.

Main Theorem. $\Pi_2\mathrm{SPACE}(n) \subseteq \Sigma_2\mathrm{SPACE}(n)$.

Our technique to prove main theorem is based on an idea due to Mahaney[4] and some new ideas. At the beginning of proving main theorem, we define some notions and show several basic lemmas about those notions.

Let $\langle\cdot,\cdot\rangle$ denotes a pairing function on $\{0,1\}^*$ satisfying the conditions that $\langle x,y\rangle$ is computable within space $O(|x|+|y|)$ for each $x,y \in \{0,1\}^*$, its inverse functions (both left and right) is computable within space $O(|\langle x,y\rangle|)$, and there is a constant $c \geq 1$ such that for each $x,y \in \{0,1\}^*$, $|\langle x,y\rangle|$ is bounded above by $c(|x|+|y|)$. In this section, both 1^0 and 0^0 denote the empty word conventionally.

Definition 1. For a nonnegative integer k, bin(k) denotes the ordinary binary notation of k. For a language $A \subseteq \{0,1\}^*$, ex(A) (called the extension of A) and pc(A) (called the pseudo-complement of A) are defined as follows: $\mathrm{ex}(A)=1^*0 \cup \{ x1 \mid x \text{ is in } A \}$, and $\mathrm{pc}(A) = \{ 1^n0\langle x,\mathrm{bin}(k)\rangle \mid x \text{ in } \{0,1\}^*, k,n \geq 0, |x| \leq n,$ there are $x_1,\ldots,x_k$ such that each x_i's are pairwise distinct and for $1 \leq i \leq k$, $|x_i| \leq n$, $x_i \neq x$ and x_i is in $A \}$. Intuitively speaking, $1^n0\langle x,\mathrm{bin}(k)\rangle$ is in pc(A) iff $|x| \leq n$ and we can choose k distinct

words of length at most n from A-{x}. Furthermore, the census of A, denoted by C_A, is a function on natural numbers defined as $C_A(n)=|\{ x \text{ in } A \mid |x|\leqslant n\}|$ for each $n\geqslant 0$.

Lemma 1. Let $A\subseteq\{0,1\}^*$ be a language in NSPACE(n). Then, both ex(A) and pc(A) are in NSPACE(n).

proof. It is obvious that ex(A) is in NSPACE(n). We prove only that pc(A) is in NSPACE(n). Let M be a linear space-bounded NTM which accepts A. Then, we construct a linear space-bounded NTM as follows.

```
input 1^n0<x,bin(k)> ;
if  |x|>n or k≥2^(n+1)  then reject and halt ;
  else  if k=0 then accept and halt ;
k_1 <- 0 ;  y <- @  ;
while  k_1<k
  do  guess a y' such that |y'|≤n and y ⊏ y' ;
        if y'=x then  reject and halt ;
        simulate M on y' ;
        if M reaches the accepting state
          then  y <- y' ; k_1 <- k_1+1 ;
          else  reject and halt ;
  od ;
accept and halt.
```

In the above, $\sqsubset$ denotes a suitable linear ordering on $\{0,1\}^* \cup \{@\}$ satisfying that $@ \sqsubset x$ for any $x \in \{0,1\}^*$. The proof of the

correctness of the above machine is left to the readers.

Lemma 2. For any $A \subseteq \{0,1\}^*$, each $k,n \geq 0$ and each x in $\{0,1\}^*$ satisfying $|x| \leq n$, the followings hold.

(1) $1^n0\langle x,bin(k)\rangle \notin pc(A)$ if $k > C_A(n)$,

(2) $1^n0\langle x,bin(k)\rangle \in pc(A)$ if $k < C_A(n)$, and

(3) $1^n0\langle x,bin(k)\rangle \in pc(A) \iff x \notin A$ if $k = C_A(n)$.

proof. (1) Assume that $k > C_A(n)$. Then, we cannot choose k distinct words of length at most n from A. Hence, $1^n0\langle x,bin(k)\rangle$ is not in $pc(A)$.

(2) Assume that $k < C_A(n)$. Then, we can choose k distinct words of length at most n from $A-\{x\}$. Hence, $1^n0\langle x,bin(k)\rangle$ is in $pc(A)$.

(3) Assume that $k = C_A(n)$. If x is not in A, then we can choose k distinct words of length at most n from A $(=A-\{x\})$, and hence $1^n0\langle x,bin(k)\rangle$ is in $pc(A)$. If x is in A, then we cannot choose k distinct words of length at most n from $A-\{x\}$ (such a choice must always contains x), and hence $1^n0\langle x,bin(k)\rangle$ is not in $pc(A)$.

Lemma 3. For any $A \subseteq \{0,1\}^*$ and each $n \geq 0$, $C_{ex(A)}(n+1) = C_A(n)+n+1$.

proof Obvious from Definition 1.

Lemma 4. For any $A \subseteq \{0,1\}^*$, $k,n \geq 0$, $k = C_A(n)$ if and only if $1^{n+1}0\langle 1^n0,bin(k+n+1)\rangle \notin pc(ex(A))$ and $1^{n+1}0\langle 1^n0,bin(k+n)\rangle \in pc(ex(A))$.

proof. In the proof of this lemma, we apply Lemma 2 for ex(A) instead of A.

(=>) Assume $k=C_A(n)$. Since 1^n0 is in ex(A) and $k+n+1 = C_{ex(A)}(n+1)$ from Lemma 3, $1^{n+1}0<1^n0,bin(k+n+1)> = 1^{n+1}0<1^n0,bin(C_{ex(A)}(n+1))>$ is not in pc(ex(A)) from Lemma 2(3). On the other hand, since $k+n<C_{ex(A)}(n+1)$, $1^{n+1}0<1^n0,bin(k+n)>$ is in pc(ex(A)) from Lemma 2(2).

(<=) Assume that the right hand holds and $k\neq C_A(n)$. Then, we consider three cases below.

Case 1: $k<C_A(n)$. In this case, $k+n+1<C_{ex(A)}(n+1)$. Hence, $1^{n+1}0<1^n0,bin(k+n+1)>$ is in pc(ex(A)) from Lemma 2(2).

Case 2: $k-1>C_A(n)$. In this case, $n+k>C_{ex(A)}(n+1)$. Hence, $1^{n+1}0<1^n0,bin(k+n)>$ is not in pc(ex(A)) from Lemma 2(1).

Case 3: $k-1=C_A(n)$. In this case, $k+n=C_{ex(A)}(n+1)$, and also, 1^n0 is in ex(A). Hence, $1^{n+1}0<1^n0,bin(k+n)>$ is not in pc(ex(A)) from Lemma 2(3).

In each cases, a contradiction occurs. Hence, $k=C_A(n)$ if the right hand holds.

From Lemma 4, we can construct an efficient algorithm for computing the census of each language in NSPACE(n).

Definition 2. For a language $A\subseteq\{0,1\}^*$, define $cen(A) = \{ 1^n0bin(k) \mid n,k\geq 0,\ k=C_A(n) \}$.

Lemma 5. For any $A \subseteq \{0,1\}^*$, if A is in NSPACE(n) then cen(A) is in Σ_2SPACE(n).

proof. From Lemma 1, pc(ex(A)) is in NSPACE(n). Let M_1 be an O(n) space-bounded NTM which accepts pc(ex(A)), and let M_2 be an O(n) space-bounded π_1TM which accepts $pc(ex(A))^c$. Then, we construct an algorithm as follows.

```
input 1^n0bin(k), where n,k≥0 ;
if k≥2^{n+1} then reject and halt ;
k_1 <- k+n ;
k_2 <- k+n+1 ;
simulate M_1 on input 1^{n+1}0<1^n0,bin(k_1)> ;
if M_1 reaches the accepting state
  then simulate M_2 on input 1^{n+1}0<1^n0,bin(k_2)> ;
       if M_2 reaches the accepting state
         then accept else reject
  else reject.
```

It is obvious that the above algorithm can be realized by an O(n) space-bounded σ_2TM. Also, it is easy to see that from Lemma 4, the above algorithm accepts $1^n0bin(k)$ iff $k=C_A(n)$.

Now, we show the main theorem in this paper.

Proof of Main Theorem. Let M be an O(n) space-bounded π_2TM. We suppose that M's tape alphabet is $\{a_1,\ldots,a_s\}$ and M is cn+c

space-bounded, where c is a positive constant depending only on M. We also suppose that M has r states, denoted by $\{q_1,\ldots,q_r\}$. Let I be an ID of M on input of length n. Then, we encode I into a binary word $I^{\#}$ as follows. $I^{\#} = \tilde{q}_1\ldots\tilde{q}_r h_1 x_1 \ldots h_m x_m$ $(m \leqslant c|x|+c)$, where $I^{\#}$ satisfies the following conditions.

(1) Each $\tilde{q}_i$ is in $\{0,1\}$, each h_i is in $\{0,1\}$, and each x_i is in $\{ 0^i 1 0^{s-i-1} \mid 0 \leqslant i \leqslant s-1 \}$.

(2) If the state of I is q_j then $\tilde{q}_j=1$ and the others are all 0.

(3) Each h_i indicates the position of M's tape head as follows. If $h_i = 1$, then M's tape head scans the i-th tape cell; otherwise, the head does not scan this cell. Futhermore, the only one of each h_i's is one and the others are all zero.

(4) Each x_i indicates a content of the i-th tape cell as follows. If the j-th bit of x_i is 1, then the content of the i-th cell is a_j. We note that the form of each x_i is restricted in (1).

(5) $|I^{\#}|>r$ and $|I^{\#}|-r \equiv 0 \pmod{(s+1)}$.

If a binary word $I^{\#}$ satisfies the above conditions, then we call $I^{\#}$ a valid encoding of an ID I of M. We define a language $L\langle M\rangle' = \{ I^{\#} \mid I^{\#}$ is a valid encoding of an existential ID I of M, and there is a computation path of M from I to an accepting ID whose internal IDs are all existential and the length of each encoding word of whose internal IDs is equal to $|I^{\#}| \}$. Obviously, $L\langle M\rangle$ is in NSPACE(n)(actually, we construct an O(n) space-bounded NTM from M which accepts $L\langle M\rangle$). Then, the following facts hold from the definition of $L\langle M\rangle$ and Lemma 2(3) (also, recall the assumptions for alternating Turing machines in section 2).

Fact 1. Let exID(M,x) be the set of existential IDs of M on input x and let I_0 denote the initial ID of M on x. Then, M accepts x if and only if for each I in exID(M,x) such that there is a computation path from I_0 to I whose internal IDs are all universal, $I^{\#}$ is in L<M>.

Fact 2. Let $I^{\#}$ be a valid encoding of an existential ID of M on input of length n. Then, $I^{\#}$ is in L<M> if and only if $1^m 0 \langle I^{\#}, \mathrm{bin}(C_{L<M>}(m))\rangle$ is not in pc(L<M>), where $m=(s+1)(cn+c)+r$.

From the fact L<M> is in NSPACE(n) and Lemma 1, the above fact 2 tells us that given a correct census of L<M>, we can construct an O(n) space-bounded π_1TM which accepts L<M>.

Now, we construct an algorithm which accepts the same language as M. Intuitively speaking, the algorithm operates as follows. Given an input of length n, it first guesses a census of L<M> up to length $m=(s+1)(cn+c)+r$. Next, it checks the correctness of the guessed census as in Lemma 5. If the census is correct, then it begins to simulate M until an existential ID occurs. Finally, if an existential ID occurs, then it decides whether that ID is in L<M> by simulating a π_1TM in Fact 2.

Let $M_{\overline{pc}}$ be an O(n) space-bounded π_1TM which accepts $pc(L<M>)^c$, and let M_{cen} be an O(n) space-bounded σ_2TM which accepts cen(L<M>). Since L<M> is in NSPACE(n), such machines

exist from Lemma 1 and Lemma 5. Then, a desired algorithm is as follows.

input x ;
m <- (s+1)(c|x|+c)+r ;
guess a binary word bin(k) such that |bin(k)|$\leq$m+1 ;
simulate M_{cen} on input $1^m 0 bin(k)$;
if M_{cen} does not reach the accepting state
  then reject
  else simulate M on input x until an existential ID I occurs ;
    if no existential ID occur
      then if M reaches the accepting state
        then accept **else** reject
      else simulate $M_{\overline{pc}}$ on input $1^m 0 \langle I^{\#}, bin(k) \rangle$;
        if $M_{\overline{pc}}$ reaches the accepting state
          then accept **else** reject.

It is easy to see that the above algorithm can be realized by an O(n) space-bounded Σ_2TM. We show the correctness of the algorithm below. Assume the algorithm accepts an input x. Then, there is a nonegative integer k such that M_{cen} accepts $1^m 0 bin(k)$ and for each I in exID(M,x), if there is a computation path from I_0 to I whose internal IDs are all universal, then $M_{\overline{pc}}$ accepts $1^m 0 \langle I^{\#}, bin(k) \rangle$, where m=(s+1)(c|x|+c)+r and I_0 denotes the initial ID of M on x. Since $k=C_{L\langle M \rangle}(m)$ from the fact M_{cen} accepts $1^m 0 bin(k)$, for each I in exID(M,x), if there is a computation path from I_0 to I whose internal IDs are all universal, then $M_{\overline{pc}}$

accepts $1^m 0 \langle I^{\#}, \mathrm{bin}(C_{L\langle M\rangle}(m))\rangle$, and hence, $I^{\#}$ is in $L\langle M\rangle$ from Fact 2. Thus, M accepts x from Fact 1. The inverse direction is similar. This complete the proof.

Corollary 1. $\Sigma_2\mathrm{SPACE}(n) = \Pi_2\mathrm{SPACE}(n)$.

Corollary 2. $\Sigma_2\mathrm{SPACE}(n) = \Sigma_k\mathrm{SPACE}(n)$ for any $k \geqslant 2$.

Corollary 3. $\Sigma_2\mathrm{SPACE}(S(n)) = \Sigma_k\mathrm{SPACE}(S(n))$ for any space-constructible $S(n) \geqslant n$.

References

1. A.K. Chandra, D.C. Kozen, and L.J. Stockmeyer, Alternation, J. Assoc. Comput. Mach. 28(1981), 114-133.

2. J.E. Hopcroft and J.D. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison-Weslay, Readings(1979).

3. S.Y. Kuroda, Classes of languages and linear bounded automata, Information and Control 7(1964), 207-223.

4. S. Mahaney, Sparse complete sets for NP: solution of a conjecture of Berman and Hartmanis, J. Comput. System Sci. 25(1982), 130-143.

5. L.J. Stockmeyer, The Polynomial-time Hierarchy, Theor. Comput. Sci. 3(1976), 1-22.